CSE 559A: Computer Vision

[credit: danjodon.deviantart.com]

Fall 2017: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Staff: Abby Stylianou (abby@wustl.edu), Jarett Gross (jarett@wustl.edu)

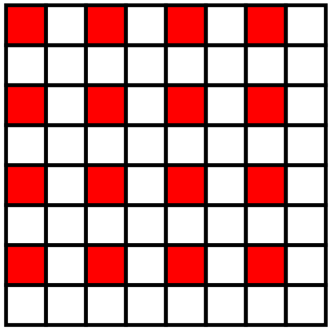
http://www.cse.wustl.edu/~ayan/courses/cse559a/

Sep 12, 2017

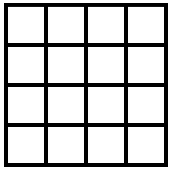
ADMINISTRIVIA

- Homework becomes available at 5pm today.
- Posted on blackboard. Submission also on blackboard.
- If you are sitting in and would like a copy, e-mail me. (won't be graded)
- Lots of support code (you fill in specific functions). Read the support code !
- $\LaTeX$ template for report is now in the resources section, along with tutorials/links.
- Please finish information section (discussions, online sources, number of hours spent)

SCALE & ALIASING



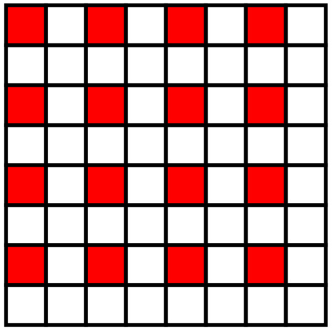
W x H



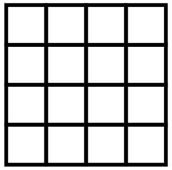
(W/2) x (H/2)

"Resize" Images

SCALE & ALIASING

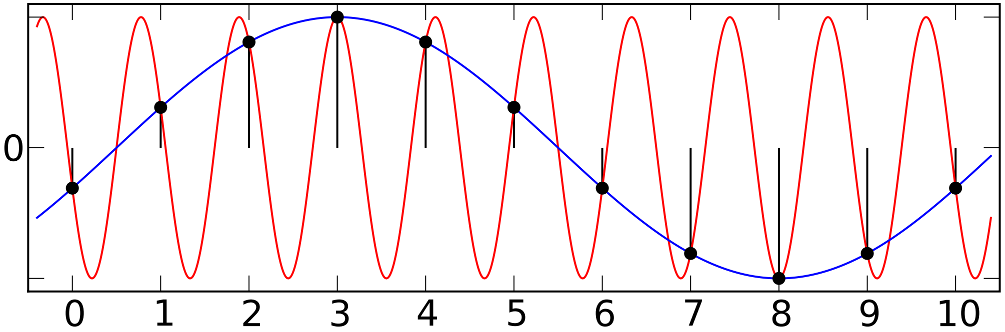


W x H



(W/2) x (H/2)

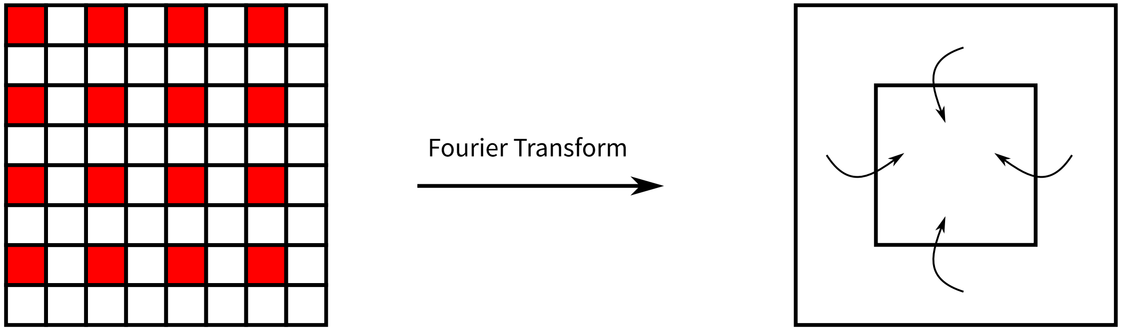
"Resize" Images



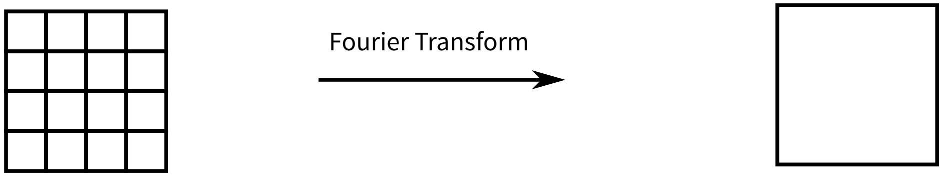
[Source: Wikipedia]

"Aliasing"

SCALE & ALIASING

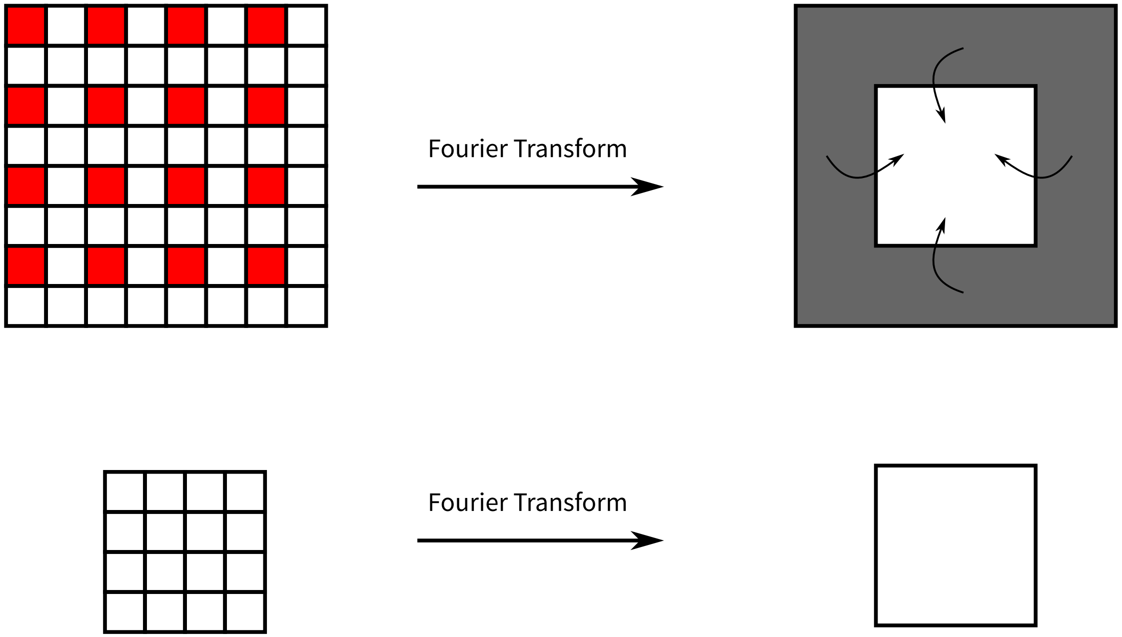


If you write it out, you see the higher freq. components get folded into lower freq.



Remember, in the two cases F[u,v] is defined with respect to different width and height W_x and H_x, and for different ranges of (u,v).

SCALE & ALIASING



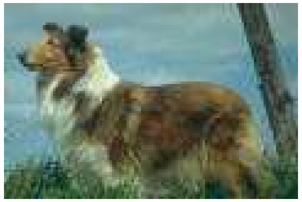
Make sure there are
no high frequencies
before sub-sampling !

Low-pass filter, i.e.,
Smooth Image before
sub-sampling.

SCALE & ALIASING

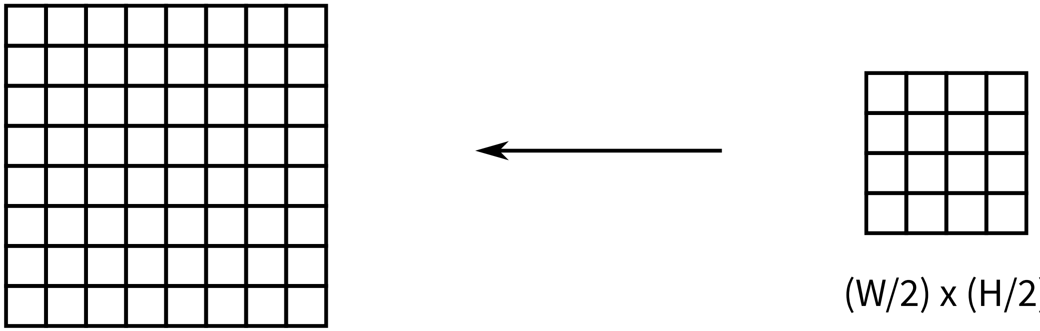
Without Smoothing

With Smoothing



Sometimes the camera itself makes aliased measurements: if spatial sensitivity is low at edges of pixel.

SCALE & ALIASING



(W/2) x (H/2)

W x H

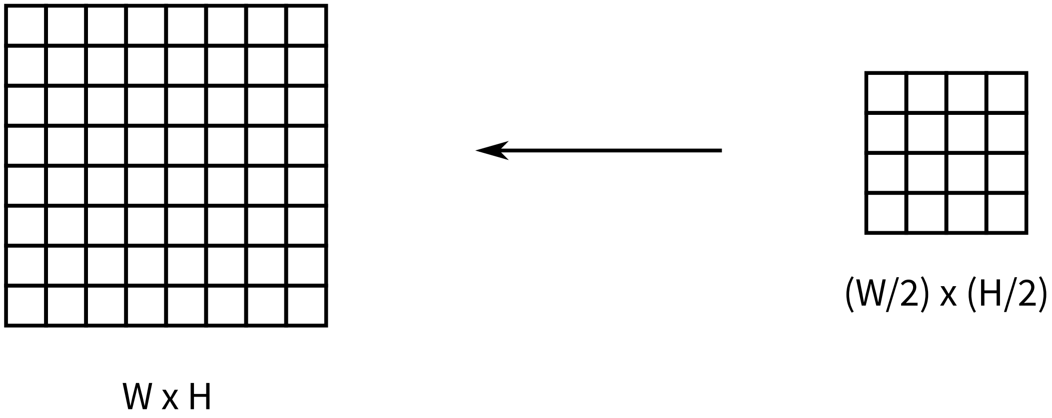
"Resize" Images

- Need to hallucinate missing information.
- Lots of research (super-resolution).

- Simplest Approach: Nearest neighbor

$$Y[n] = X[\text{round}(n/2)]$$

SCALE & ALIASING

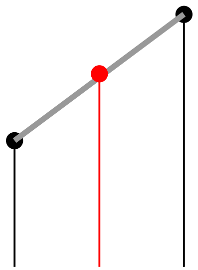


"Resize" Images

- Need to hallucinate missing information.
- Lots of research (super-resolution).

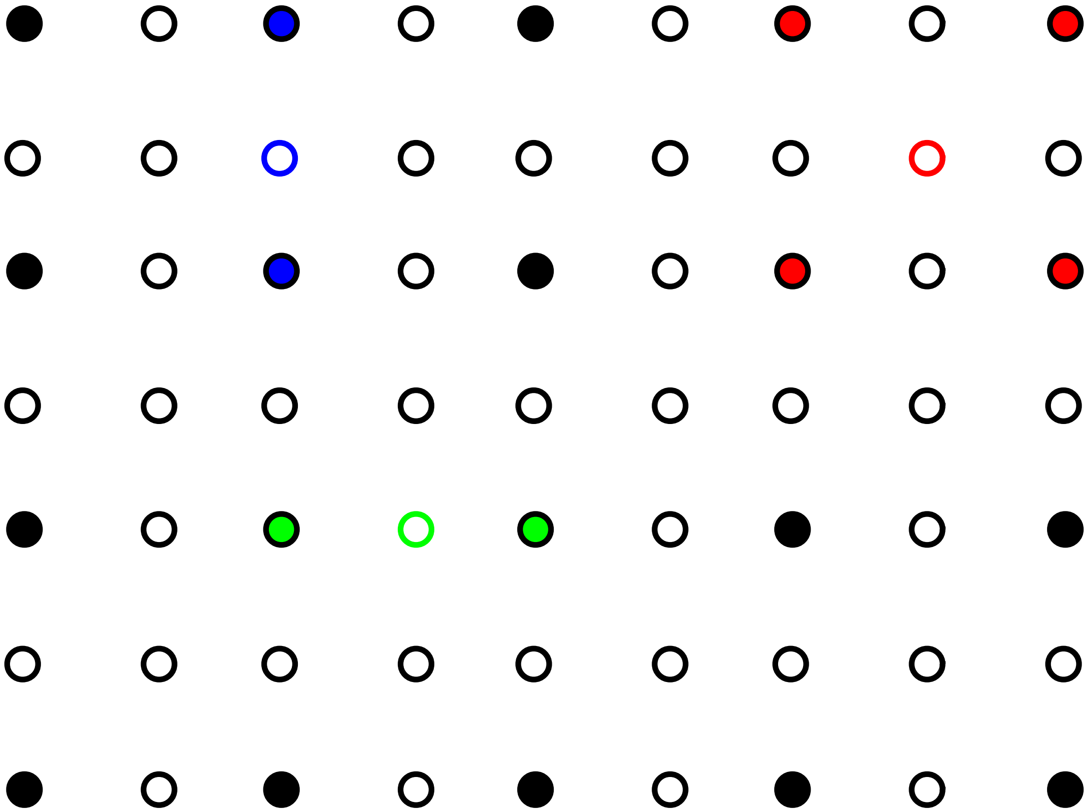
- Simplest Approach: Nearest neighbor

- Simple Approach: (Bi) Linear Interpolation



For up-sampling by 2 in 1-D, missing values are just the average of the left and right present values.

SCALE & ALIASING



1/4	1/2	1/4
1/2	1	1/2
1/4	1/2	1/4

Can achieve this by filling with zeros, and convolution with a 3x3 kernel.

EFFICIENT COMPUTATION

- Convolution, in the most general case, takes $O(n_x n_k)$ time.
  - $n_x = W_x H_x, n_k = W_k H_k$.
- Convolution in the frequency domain:
  - FFT, point-wise multiply, Inverse FFT
  - FFT/IFFT complexity is $O(n_x \log n_x)$ (Most efficient for power of 2 image size)
  - May be worth it for large kernels
  - Or same image convolved with many different kernels

EFFICIENT COMPUTATION

Separable Kernels

$$G[n_x, n_y] \propto \exp\left(-\frac{n_x^2 + n_y^2}{2\sigma^2}\right) = G_x[n_x]G_y[n_y]$$

- $x-$ and $y-$ derivatives of Gaussian also separable.

- Realize that $k[n_x, n_y] = k_x[n_x]_y k[n_y] = k_x *_{\text{full}} k_y$.

  This is by interpreting k_x and k_y as having size $W_k \times 1$ and $1 \times H_k$.

- So $X * k = X * (k_x * k_y) = (X * k_x) * k_y$ This takes $W_k + H_k$ operations instead of $W_k H_k$.

- Often if a kernel itself isn't separable, it can be sometimes expressed as a sum of separable kernels.
- E.g., Unsharp Mask: $(1 + \alpha)\delta - \alpha G_\sigma$ (don't combine!)
- Could also try to do this automatically using SVD.

EFFICIENT COMPUTATION

Recursive Computation

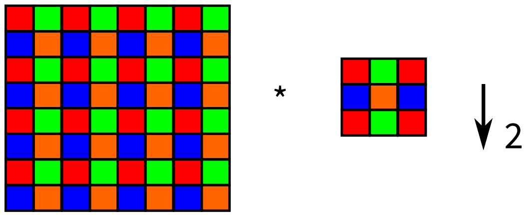
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

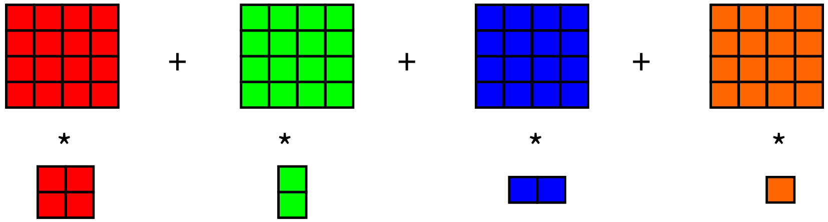
- Sometimes can decompose into convolution with sparse kernels.
- Many implementations of `convolve2d` won't make use of sparsity.
    - But you can write your own.

EFFICIENT COMPUTATION

Smooth and Sub-sample

- Don't smooth and subsample !
- For sub-sampling by two, you're computing 4x as many smooth filter outputs as you need to.





- Similarly, using zero-filling + convolution for upsampling is inefficient.

EFFICIENT COMPUTATION

numpy Specifics

1. In general, prefer algorithms that have lower total number of multiplies / adds.
2. Try to use `scipy.signal.convolve2d` (subject to rule 1). It is optimized for cache reads, parallel execution, etc.
   (I import it often as: `from scipy.signal import convovle2d as conv2`)
3. Similarly, avoid `for` loops and use element-wise operations on large arrays, matrix multiplies (`np.matmul` / `np.dot`), etc.

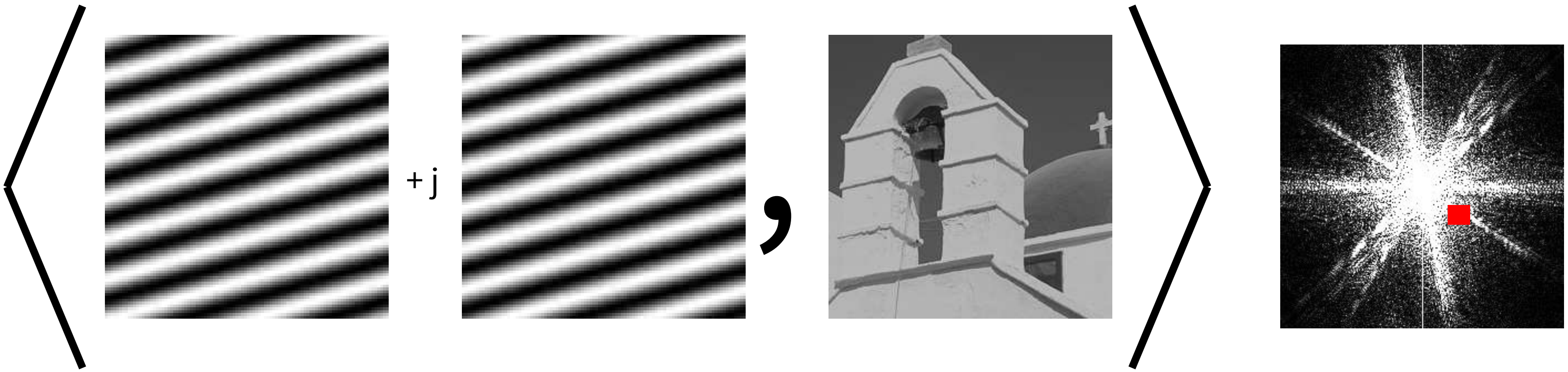
- Some of these things are faster in python because a single large operation runs natively instead of returning to the compiler. But they're also faster because these operations are often 'atomics' in lower-level libraries too (BLAS), and have been highly optimized for modern hardware.
- Thinking about designing your algorithm in terms of these atomic operations is useful beyond python.
- Some points in problem sets allocated for efficient code.

MULTI-SCALE REPRESENTATIONS

Fourier Transform useful for:

Diagonalizing Convolution $A_k = S D_k S^*$

Concentrating energy (high manitudes) in fewer coefficients



But is the FT interpretable ? Does it tell us something about the image ?

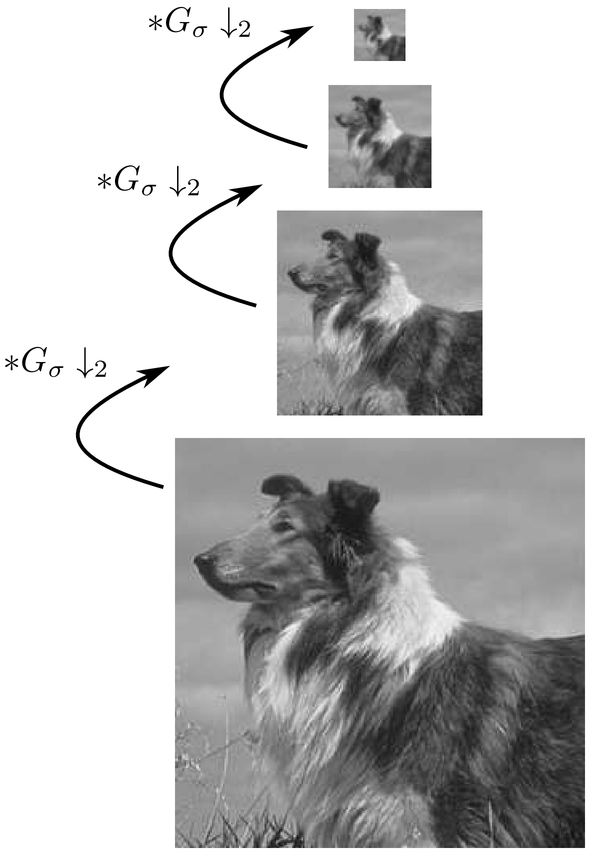
MULTI-SCALE REPRESENTATIONS

- $F[u, v]$ is intuitively average variation in image at that frequency.
- But averaged across the entire image.
- This isn't useful because images aren't "stationary"
- Different parts of the image, "have different frequencies".

- FT decomposition of different levels (coarse/fine) of variation: but without sense of spatial location.
- Multi-scale representations aim to address this.

MULTI-SCALE REPRESENTATIONS

Gaussian Pyramid

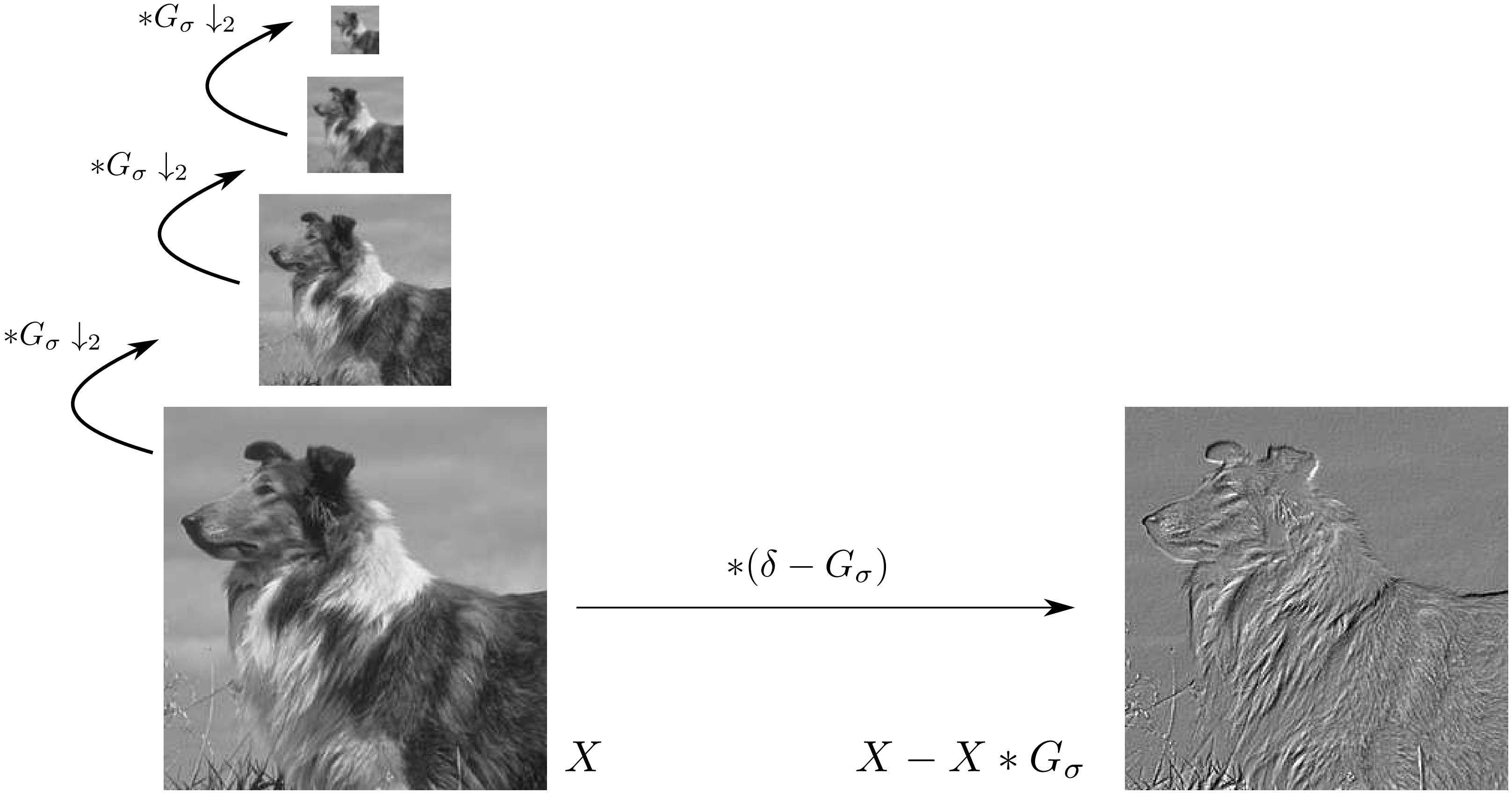


Useful for analyzing image at multiple scales.

E.g., apply the same method (edge detection / CNN) at multiple levels of the pyramid.

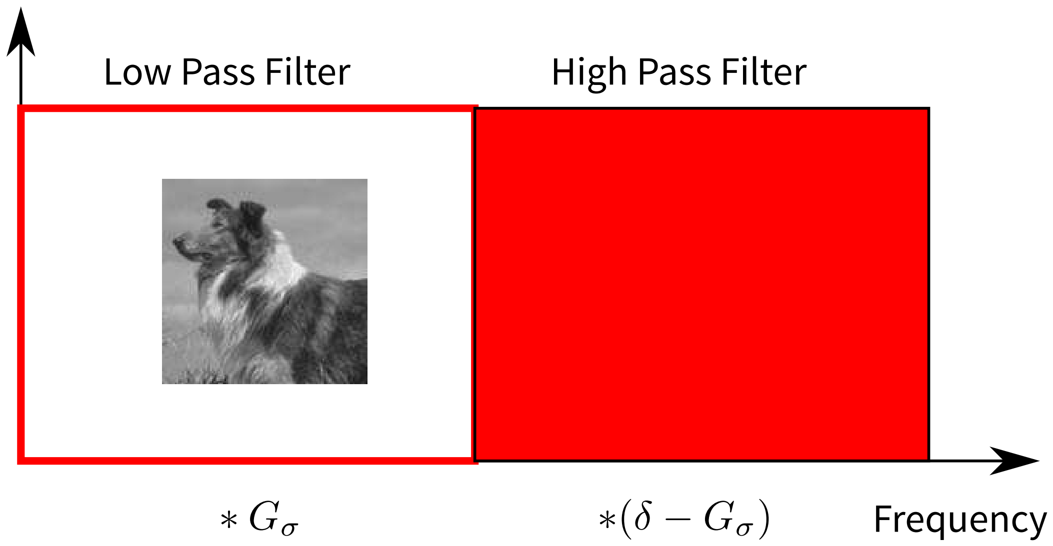
MULTI-SCALE REPRESENTATIONS

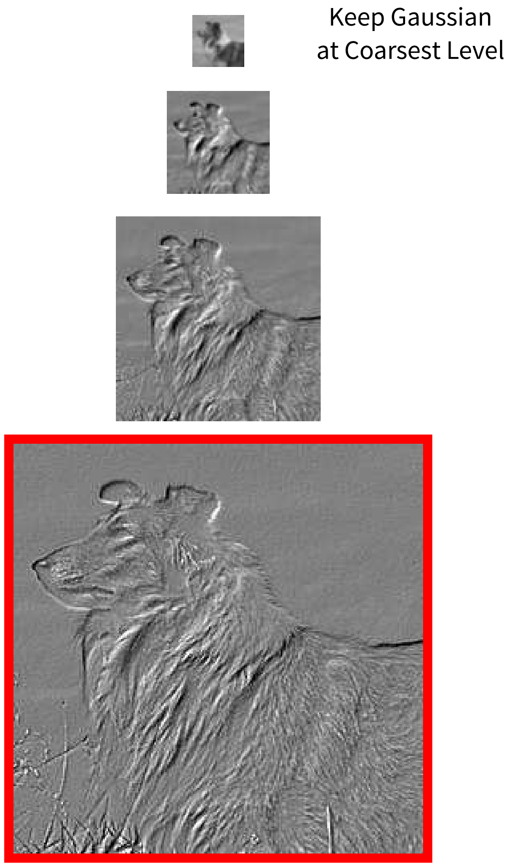
Laplacian Pyramid



MULTI-SCALE REPRESENTATIONS

Laplacian Pyramid

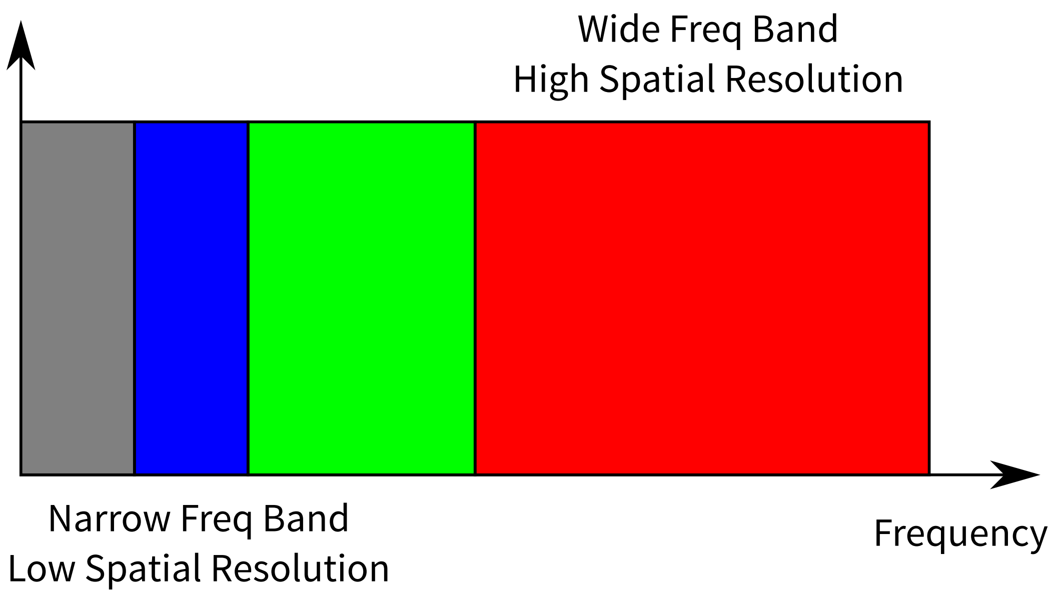


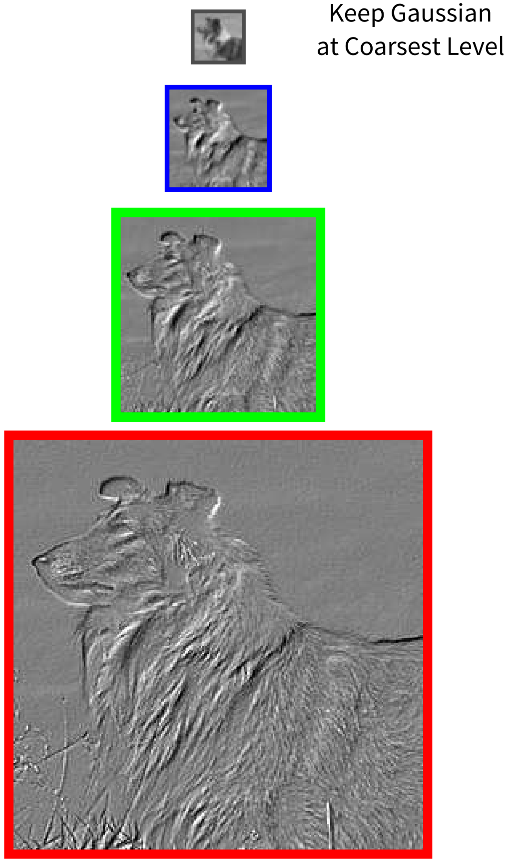


MULTI-SCALE REPRESENTATIONS

Laplacian Pyramid

No orientation selectivity: See Steerable Pyramids
http://www.cns.nyu.edu/~eero/steerpyr/





MULTI-SCALE REPRESENTATIONS

Wavelet Pyramid

Gaussian & Laplacian pyramids are good for analysis, but not really a representation.
Not easy / possible to go from pyramid coefficients back to original image (like Fourier Transform).

Consider a 2x2 Pixel Block

a	c
b	d

$$L = \frac{a+b+c+d}{2} \qquad H_2 = \frac{c+d-a-b}{2}$$

$$H_1 = \frac{b+d-a-c}{2} \qquad H_3 = \frac{a+d-b-c}{2}$$

$$\begin{bmatrix} L \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

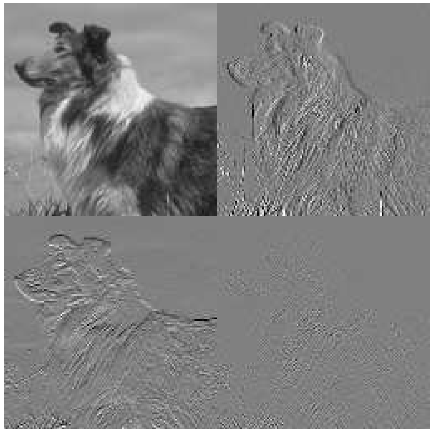
Unitary Matrix / Co-ordinate transform

MULTI-SCALE REPRESENTATIONS

Wavelet Pyramid

Can be achieved by
Convolution + Downsample

Divided by 2
for Visualization



For every non-overlapping
2x2 Patch in the Image

Still a Co-ordinate Transform !

a	c
b	d

$$L = \frac{a+b+c+d}{2}$$

$$H_2 = \frac{c+d-a-b}{2}$$

$$H_1 = \frac{b+d-a-c}{2}$$

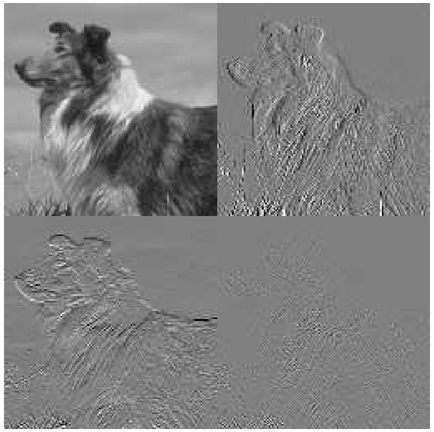
$$H_3 = \frac{a+d-b-c}{2}$$

MULTI-SCALE REPRESENTATIONS

Wavelet Pyramid

$$\begin{bmatrix} L \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Can Invert
to get Image



a	c
b	d

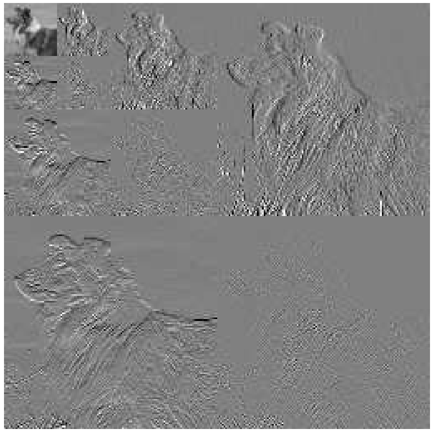
$$L = \frac{a+b+c+d}{2} \qquad H_2 = \frac{c+d-a-b}{2}$$

$$H_1 = \frac{b+d-a-c}{2} \qquad H_3 = \frac{a+d-b-c}{2}$$

MULTI-SCALE REPRESENTATIONS

Wavelet Pyramid

Apply Recursively



Wavelet Transform

a	c
b	d

$$L = \frac{a+b+c+d}{2} \qquad H_2 = \frac{c+d-a-b}{2}$$

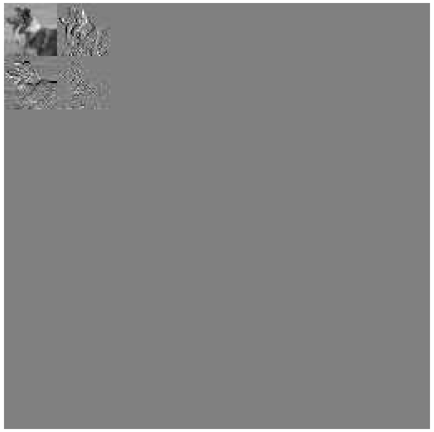
$$H_1 = \frac{b+d-a-c}{2} \qquad H_3 = \frac{a+d-b-c}{2}$$

"Harr"

Others based on different filters

MULTI-SCALE REPRESENTATIONS

Wavelet Pyramid



Applications: Analysis, image modeling & restoration, compression

IMAGE RESTORATION

- Recovering "true" image $X[n]$ from observed image $Y[n]$
  (can apply to image like objects too, like depth maps)
- Step 1: Assume we know the degradation model, or the mapping from X to Y

$$p(Y|X): \quad \text{Distribution of possible } Y\text{s that we can get from } X$$

Example: Additive White Gaussian Noise

$$Y[n] = X[n] + \epsilon[n], \quad \epsilon[n] \sim \mathcal{N}(0, \sigma^2)$$

$$p(Y[n]|X[n]) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(Y[n]-X[n])^2}{2\sigma^2}\right) \Rightarrow p(Y|X) = \prod_n \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(Y[n]-X[n])^2}{2\sigma^2}\right)$$

$$p(Y|X) = (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{\|Y-X\|^2}{2\sigma^2}\right), \quad N = \text{Total no. of pixels.}$$

Example: Blur (+ noise)

$$Y[n] = (X * k)[n] + \epsilon[n], \quad \epsilon[n] \sim \mathcal{N}(0, \sigma^2)$$

For fronto-parallel scenes, defocus and (parallel) motion blur can be modeled as convolution.

IMAGE RESTORATION

Bayesian View

- Step 2: Define a prior distribution $p(X)$, which encodes your *a-priori* knowledge about statistics of natural images.
- Bayes Rule: Given observation likelihood and prior, gives us *Posterior distribution*

$$p(X|Y) = \frac{P(Y|X)P(X)}{\int_{X'} P(Y|X')P(X')dX'} \propto P(Y|X)P(X)$$

- Could estimate X as the mean / mode of this distribution. Let's focus on the maximum. Called the Maximum A Posteriori (MAP) estimate:

$$\hat{X} = \arg\max_X P(Y|X) = \arg\max_X P(Y|X)P(X)$$

$$= \arg\min_X - \log P(Y|X) - \log P(X)$$

IMAGE RESTORATION

- For denoising:

$$\hat{X} = \arg\min_X \frac{\|Y - X\|^2}{2\sigma^2} - \log P(X)$$

- For deblurring:

$$\hat{X} = \arg\min_X \frac{\|Y - A_k X\|^2}{2\sigma^2} - \log P(X)$$

Instead of Bayesian view, can think of minimizing a data cost + 'regularizer' on X:

$$\hat{X} = \arg\min_X \frac{\|Y - X\|^2}{2\sigma^2} + R(X)$$

$$\hat{X} = \arg\min_X \frac{\|Y - A_k X\|^2}{2\sigma^2} + R(X)$$

IMAGE RESTORATION

Now let's say we were doing denoising, and our regularizer / prior was pixel-wise.

$$R(X) = \sum_n R_n(X[n])$$

We could find the estimate of each $X[n]$ independently.

$$\hat{X}[n] = \arg\min_x \frac{(Y[n]-x)^2}{2\sigma^2} + R_n(x), \quad \forall n$$

For example, $R_n(x) = \frac{(x-0.5)^2}{2\sigma_x^2}$ (- log probability for $\mathcal{N}(0.5, \sigma_x^2)$)

$$X[n] = \arg\min_x \frac{(Y[n]-x)^2}{2\sigma^2} + \frac{(0.5-x)^2}{2\sigma_x^2}$$

$$= \frac{Y[n]\sigma_x^2 + 0.5\sigma^2}{\sigma_x^2 + \sigma^2}$$

(Take derivative of cost function set to 0, check second derivative is positive)

IMAGE RESTORATION

But what happens when the function doesn't decompose pixel-wise ?

Example: Regularizer puts quadratic penalty on gradients.

$$R(X) = \frac{\lambda}{2} \sum_n (G_x * X)[n]^2 + (G_y * X)[n]^2$$

$$\hat{X} = \arg\min_X \frac{1}{2\sigma^2} \|Y - X\|^2 + \frac{\lambda}{2} \left[\|A_{gx} X\|^2 + \|A_{gy} X\|^2 \right]$$

where A_{gx} and A_{gy} are matrix form of convolution with x- and y- derivative filters.

Expanding this out:

$$\hat{X} = \arg\min_X X^T \, Q \, X - 2 \, X^T \, B + c$$

where:

$$Q = \frac{1}{2\sigma^2} I + \frac{\lambda}{2} \left(A_{gx}^T A_{gx} + A_{gy}^T A_{gy} \right), \quad \text{where I is } N \times N \text{ identity matrix.}$$

$B = Y/\sigma^2$, and c is independent of X.

IMAGE RESTORATION

$$\hat{X} = \arg\min_{X} X^T\, Q\, X - 2\, X^T\, B + c$$

where:

$$Q = \frac{1}{2\sigma^2} I + \frac{\lambda}{2}\left(A_{gx}^T A_{gx} + A_{gy}^T A_{gy}\right), \quad \text{where I is } N \times N \text{ identity matrix.}$$

$$B = Y/\sigma^2, \text{ and } c \text{ is independent of } X.$$

- This is a vector-equivalent of a quadratic form. . . .

- We can find X by computing derivative of the cost wrt X, and setting it to 0.

$$2\,\hat{Q}X - 2\,B = 0 \Rightarrow \hat{X} = Q^{-1}B$$

- All eigen-values of Q are strictly above 0. (Why ?)
- "Second" derivative of the cost along any direction of X is positive. So minima, not maxima.
- Q is called a symmetric positive-definite matrix. The cost function is convex along all directions of X.

The problem: Q is $N \times N$, where N is the number of pixels !

IMAGE RESTORATION

Useful Reading:

- https://en.wikipedia.org/wiki/Positive-definite_matrix
- https://en.wikipedia.org/wiki/Matrix_calculus
  We'll be using the "denominator layout" convention.